# MINERVA: Information-Centric Programming for Social Sensing

Shiguang Wang, Shaohan Hu, Shen Li, Hengchang Liu, Md Yusuf Sarwar Uddin, Tarek Abdelzaher
Department of Computer Science, University of Illinois at Urbana-Champaign
Email: {swang83, shu17, shenli3, hl4d, mduddin2, zaher}@illinois.edu

***Abstract*—In this paper, we introduce Minerva; an *information-centric* programming paradigm and toolkit for social sensing. The toolkit is geared for smartphone applications whose main objective is to collect and share information. Information-centric programming refers to a publish-subscribe paradigm that maximizes the amount of information delivered. Unlike a traditional publish-subscribe system where publishers are assumed to have independent content, Minerva is geared for social sensing applications where different sources (participants sharing sensor data) often overlap in information they share. For example, through lack of coordination, they might collect redundant pictures of the same scene or redundant speed measurements of the same street. The main contribution of Minerva, therefore, lies in a data prioritization scheme that maximizes information delivery from publishers to subscribers by reducing redundancy, taking into account the *non-independent* nature of content. The algorithm is implemented on Android phones on top of the recently introduced named data networking framework. Evaluation results from both two smartphone-based experiments and a large-scale real data driven simulation demonstrate that the prioritization algorithm outperforms other candidates in terms of information coverage.**

## I. Introduction

This paper introduces Minerva; a novel publish-subscribe-based programming system for optimizing information throughput in social sensing applications. Social sensing refers to the act of crowd-sourcing sensor data collection to volunteer participants in exchange for offering data services of interest. A common example is the collection and sharing of traffic speed data by drivers on different streets for purposes of computing speed maps that help plan individuals' commute.

We argue that development of social sensing applications calls for an *information-centric* programming paradigm in that the underlying run-time support is geared at maximizing *information flow*. This, as we show below, is not the same as maximizing *data throughput*. Social sensing applications fit a publish-subscribe model, where the sources involved in data collection are the publishers and the service that computes the quantities of interest is the subscriber.[1] Sources are typically mobile, such as phones or cars, and opportunistic WiFi offloading is used to reduce the cost of data upload (most data plans now charge for 3G/4G data upload, which makes it an unattractive choice for the sensing application). Hence, information propagates from one participant to another when they meet, and is uploaded to the subscriber when a participant has a free upload opportunity. Importantly, unlike the traditional publish-subscribe model, where publishers are independent, social sensing applications typically exhibit information overlap among sources. For example, vehicles waiting in the same traffic jam may collect very similar observations about traffic. Redundancy in data collection thus leads to inefficiency, which motivates a system that can recognize and eliminate the redundancy. Such a system would maximize information flow, as opposed to mere data throughput.

The main contribution of Minerva lies in its information-maximizing data prioritization scheme. It transmits publishers' data in an order that prioritizes non-redundant data. Hence, if data transfer is interrupted before all data are transmitted, ideally the most informative data will have been transmitted for the given transfer size. The scheme is suitable for mobile environments where connectivity between nodes may be interrupted due to the nodes' mobility patterns and limited battery capacities. We show that without knowing the data transmission time in advance, which is the common case, no prioritization scheme can guarantee the optimal information throughput. Instead, an approximation bound is derived that is achieved by our prioritization algorithm, making it provably near-optimal.

From an API perspective, Minerva separates application-specific components from application-independent components. We recognize that *information* is a measure that may mean different things to different applications. In this paper, it refers to any application-specific quantity from which the application derives its utility. To keep the information-maximization support in Minerva as application-independent as possible, we ask the programmer to define only one application-specific function per collected content type; namely, *a map function*, which takes a data object as operand and returns its position and coverage in a virtual information space, where objects that are closer to each other might have more information overlap, and vice versa. An example mapping function could be one that places the object as a point in a space whose dimensions are the location and time at data capture. Hence, data captured at closer locations and times would have more overlap in the virtual space. Except for the programmer-defined map function, the rest of Minerva information-maximization mechanisms are entirely application-independent. Based on the given logical information coverage, Minerva automatically computes an information-maximizing data transmission order for nodes that

[1]The service also makes the computed results available, but this is done using standard dissemination techniques and is not the focus of this paper.

aims to minimize logical overlap among objects delivered in any finite time interval.

Finally, Minerva is novel in exploiting the recently proposed Named Data Networking (NDN) framework [15] to maximize information delivery. Named data networking is a network paradigm where data objects are given unique names in a hierarchical name space (reminiscent of a UNIX directory structure), allowing the network to retrieve them by name. While network support for NDN abstractions is of little relevance in this paper, by giving collected data objects descriptive names in an appropriately designed name-space, we allow logical information coverages of objects to be computed automatically from their names. Hence, the computation of data transmission priorities becomes more efficient. Specifically, it obviates the need for inspecting the content at nodes other than the source in order to determine redundancy (and hence, priority). Similarity is established based on names only.[2]

We evaluate Minerva using two smart-phone-based experiments as well as a large-scale simulation using the T-Drive dataset collected by MSRA [26]. Evaluation results demonstrate that our prioritization algorithm outperforms other candidates.

The remainder of this paper is organized as follows. We compare our work with the state of the art in Section II and present the system design in Section III. We formulate and solve our problem of maximizing information coverage in Section IV. The implementation and evaluation for our proposed solution are discussed in Section V. Finally, we conclude the paper in Section VI.

## II. State of the Art

Social sensing attracts lots of attentions in research community since it was introduced in Burke et al. [6]. Examples of early services include CenWits [13], CarTel [14], BikeNet [7], PoolView [10], and GreenGPS [9]. Also a broad overview of social sensing applications is presented in Abdelzaher et al. [4].

The idea of eliminating redundancy (typically discussed in the context of static sensor networks) was addressed in prior work on social sensing as well. For example, PhotoNet [20] is a content-aware picture delivery service that tries to reduce data redundancy in resource constrained mobile networks in post-disaster scenarios. It uses computer vision algorithms to compare photos and prioritizes which ones to transmit first in order to diversify what is ultimately reported. CARE [23] considers similar scenarios in that people inside a disaster zone exchange information they collect using disrupted and opportunistically available networking. The authors try to eliminate redundant content by utilizing different computer vision algorithms to improve bandwidth usage. Both of the preview work only propose a heuristic solution, whereas our paper is the first to derive a provably near-optimal data prioritization scheme that is the main component of information maximization. The work broadly falls in the area of Information Centric Networking (ICN), investigated in recent years [11], [12], [19].

Data prioritization algorithms have been designed to handle the undeterministic meeting time and bandwidth problems in disruption tolerant networking environments. The basic idea is that, if data transfer is interrupted before all data can be transmitted, it is desirable that the most informative data would have been transmitted for the given transfer size. Different priority schemes have been proposed that maximize application-specific metrics. For example, PhotoNet [20] maximizes diversity of transmitted images. By maximizing diversity, the receiver gets the "big picture" quicker, as opposed to potentially receiving lots of pieces of some content, and none or little of other content. Diversity-maximization, however, is prone to selecting outliers that are not representative, since by definition they are the most different from mainstram data. Liu et al. proposed a QoS-heterogeneous prioritization algorithm [16], to allow data packets with deadlines to be transmitted first in order to increase the possibility of offloading them faster. Our paper is the first to target on information *coverage*. We design a novel data prioritization algorithm that is proven to maximize coverage subject to an approximation bound.

Previous efforts exist on redundancy elimination in networks including application-level [18], [24] and packet level [5] techniques. Our work complements them in that we focus on eliminating redundancy of information of data. It is also complementary to work on sensor coverage in sensor networks [21].

We exploit the Named Data Networking (NDN) framework because it offers significant simplifications in the implementation of information-centric programming. NDN is recently proposed as a future Internet architecture, introduced by Van Jacobson [15], [27]. Since then, several papers investigated aspects of this framework such as suitability to ad hoc networking [17] and naming for mobile environments [22]. As described by Van Jacobson et al. in [15], NDN adopts a publish-subscribe paradigm to enable name-based communication in the network. The communication is *pull-based*, which means that the subscriber initiates the communication by sending an interest packet to the publisher, in which the name structure of the data that the subscriber wants to pull is described. After parsing and analyzing the interest packet, the publisher sends a data to the subscriber and waits for the next interest packet. This publish-subscribe programming paradigm fits social sensing applications well, where sensors are publishers and the backend servers subscribers. Our work is the first in proposing a programming API for social sensing applications that uses the NDN framework to simplify the maximization of information coverage.

## III. System Design

This section describes the detailed system design. We first present the system model for social sensing applications, then explain the programming framework based on this model.

[2]Minerva exploits NDN framework does not mean NDN is a must to implement Minerva. The only feature of NDN used by Minerva is that "data has name", thus it can also be implemented on TCP/IP by adding a key to each data as its name.

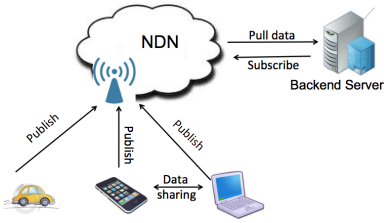



Fig. 1. System Model

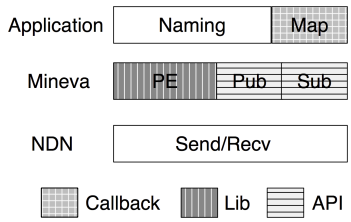



Fig. 2. Programming Framework

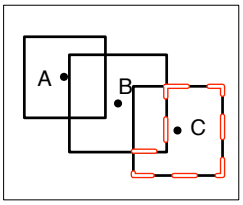



Fig. 3. Illustration of coverage and marginal coverage with 2 features.

### A. System Model

Figure 1 depicts the system model for our proposed social sensing applications. Mobile devices participate in these applications by generating and sharing sensory data, which are stored locally and uploaded to a backend server via opportunistic WiFi offloading. Hence mobile devices serve as *publishers*, and the backend server acts as the *subscriber*. Opportunistic peer-to-peer communication might also be enabled to allow information to transparently propagate from one participant to another when they meet, in hopes of finding an offloading opportunity to the server faster. We adopt the NDN framework [15], thus data generated by users are identified by descriptive names.

### B. Programming Framework

The programming support in Miverva is straightforward. Minerva provides a publish and a subscribe interface. Additionally, the application provides a callback function (one per content type), called map(), that takes as operand the name of a content object of a particular type and returns the corresponding position and coverage in a virtual information space. In the context of objects that represent sensing data, a possible map function is one that puts the object at coordinates given by the location and time at data capture, the idea being that sensor data captured at closer times and locations should, in general, be more similar and have higher coverage overlaps in the virtual space. The position and coverage of a data point are used to compute its priority in transmission that maximizes information coverage in a resource constraint environment. Objects are transmitted in the order of largest increase in marginal coverage as discussed in detail in Section IV.

As shown in Figure 2, an operational system would consist of three different layers: the application layer, the Minerva layer, and the network layer.

The application layer would take care of application-specific functions, such as content naming and publishing. Maintaining uniqueness of names is an application-specific concern not addressed in this paper. A fully specified name refers to a unique item. Names can also be partially specified to designate a collection of items that share a common name prefix. In Minerva, publishers and subscribers refer to content collections by name when expressing availability of or interest in content.

In general, applications that use our publish-subscribe system own "subdirectories" in the global name space. For example, an application called *GreenGPS* might own the subdirectory "/root/GreenGPS". The application might publish multiple types of content. Each part could start with "/root/GreenGPS/content-type". Following the content type in the name comes a listing of content attributes of relevance to the map function. A type-specific map function can therefore parse the name to determine the attributes, and compute the coordinates of the object in virtual space accordingly. For example, an object might be called "/root/GreenGPS/content-type/location/time/filename", where the location and time are the features of the data object.

In addition to the publish and subscribe functions, Minerva internally has a core function, *PE*, short for Prioritization Engine. *PE* reflects our optimization algorithm described in Section IV to compute priorities for data objects such that they are transmitted in an order that contributes to maximum coverage.

The underlying network Layer provides the standard Send/Receive functions across a network. In our implementation, we use NDN as the underlying network layer. Our solution does not require any changes to the standard NDN library (developed by PARC), thus is general enough to be compatible with other existing NDN applications.

In the next section, we describe in detail the prioritization algorithm.

## IV. Information-Maximizing Prioritization

In this section, we first introduce the definition of information coverage and formulate the information coverage maximizing problem. Then, we present the design and analysis of our algorithm.

### A. Information Coverage

Data collected in social sensing applications are not independent; they exhibit correlations from aspects such as time, space, etc. as discussed in the introduction. Thus, each data point covers a region in the *information space*, referred to as the *data coverage* as defined in Definition 1.

*Definition 1 (Data Coverage):* Suppose that there are $k$ features of the data collected in a social sensing application. The Cartesian product[3] of domains of the $k$ features forms a $k$-D *information space*. Any data point $X$ with coordinates $\langle x_1, x_2, \cdots, x_k \rangle$ covers an interval $I_j$ centered at $x_j$ on the $j$-th dimension, where $1 \leq j \leq k$. The *coverage* of $X$ is $\mathcal{C}_X = I_1 \times I_2 \times \cdots \times I_k$, where $\times$ is the Cartesian product.

[3]The Cartesian product of two sets $A$ and $B$ is a set $C$, such that $C = \{\langle x, y \rangle \mid x \in A, y \in B\}$. Similarly, we can define the Cartesian product of $k$ sets.

Please note that data of different natures and for different applications might have different coverage intervals. Thus the coverage of different data might not have the same size. By Definition 1, the coverage of a data point is a $k$-D box as illustrated in Fig. 3.

The coverage of a dataset $\mathbb{S}$ is defined as $\mathcal{C}_{\mathbb{S}} = \bigcup_{S\in\mathbb{S}} \mathcal{C}_S$. The coverage of the intersection (*resp.* union) of two datasets $\mathbb{S}_1, \mathbb{S}_2$ is defined as $\mathcal{C}_{\mathbb{S}_1\cap\mathbb{S}_2} = \mathcal{C}_{\mathbb{S}_1} \cap \mathcal{C}_{\mathbb{S}_2}$ (*resp.* $\mathcal{C}_{\mathbb{S}\cup\mathbb{S}_2} = \mathcal{C}_{\mathbb{S}_1} \cup \mathcal{C}_{\mathbb{S}_2}$).

We define the *marginal coverage* of a data point $X$ *w.r.t.* a dataset $\mathbb{S}$ in Definition 2.

*Definition 2 (Marginal Coverage):* The marginal coverage of a data point $X$ *w.r.t.* a dataset $\mathbb{S}$ is the region in the information space covered by $X$ but not covered by $\mathbb{S}$, *i.e.*, $\mathcal{MC}_{X|\mathbb{S}} = \mathcal{C}_X - \mathcal{C}_{\{X\}\cap\mathbb{S}}$.

As illustrated in Fig. 3, the area surrounded by the dashed red line is the marginal coverage of data point $C$ *w.r.t.* the dataset $\{A, B\}$. By definition, $\mathcal{MC}_{X|\emptyset} = \mathcal{C}_X$. We define the value of the coverage of a data point in Definition 3.

*Definition 3 (Coverage Value):* The coverage value of a data point $X$ in a $k$-D information space is the value of its $k$-D coverage region, defined as $\mathcal{V}(\mathcal{C}_X) = \prod_{i=1}^{k} I_i$, where $I_i$ is the coverage interval in the $i$th dimension as in Definition 1.

For example, if $k = 2$, the value of the coverage of a data point is simply the area of its coverage region in the information space. Similarly, definitions of the coverage value of a dataset and the marginal coverage value of a data point *w.r.t.* a dataset follow.

### B. Problem Definition

A common goal of social sensing is to gather information as complete as possible. One trivial solution is that when a connection establishes between two participants they sync all data, and when connecting to the backend server, a participant offloads its entire local data. However, due to the mobility and resource constraints (*e.g.*, energy), it is not always possible to sync or offload the entire dataset in a single transmission. Thus, in each transmission session, we aim to maximize the marginal information coverage value of the subset of data that can be transmitted, referred to as the MAXINFO problem, formulated below.

*Problem 1 (*MAXINFO*):* Suppose that there is a dataset $\mathbb{S}_1$ (*resp.* $\mathbb{S}_2$) on the data receiver (*resp.* the data provider). MAXINFO is to determine an order based on which the receiver should pull data from the provider such that for any data transmission size the receiver's information gain is maximized. In other words, let $\mathbb{R} \subset \mathbb{S}_2$ with cardinality $n$ is the dataset pulled by the order, then $\forall n, \forall\mathbb{T}, |\mathbb{R}| = |\mathbb{T}| = n, \mathcal{V}(\mathcal{C}_{\mathbb{S}_1\cup\mathbb{R}}) \geq \mathcal{V}(\mathcal{C}_{\mathbb{S}_1\cup\mathbb{T}})$.

Unfortunately, the unpredictability of the duration of each transmission session makes it impossible to find an order that is optimal for any $n$, which can be proved by a counter example as illustrated in Fig. 3. When $n = 1$, the optimal order is to select data $B$ first, since its coverage value is the highest. When $n = 2$, the optimal order is to select data $A$ and $C$ first, which conflicts against the optimal order when $n = 1$. However, we still need to define an optimal solution of MAXINFO in order to evaluate different solutions of the problem.

We define the optimal solution $OPT$ of MAXINFO to be an offline optimal solution with knowing the cardinality $n$ in advance. In other words, $OPT$ returns different orders with different values of $n$. And we define the approximation ratio of a solution $A$ (which is a fixed order of data points on the provider side) in Definition 4.

*Definition 4 (Approximation Ratio of* MAXINFO*):* Suppose that there is a dataset $\mathbb{S}_1$ (*resp.* $\mathbb{S}_2$) on the data requester $m_r$ (*resp.* the data provider $m_p$). A solution $A$ of MAXINFO is a fixed order to pull data from $m_p$. Let $\mathbb{A}^n \subset \mathbb{S}_2$ denote the subset of data transmitted from $m_p$ with cardinality $n$ during the transmission session. Let $\mathbb{OPT}^n$ denote the subset output by $OPT$ with $n$ known in advance. The approximation ratio of $A$ is

$$\tau = \min_{\forall n, 0\leq n\leq|\mathbb{S}_2|} \frac{\mathcal{V}(\mathcal{C}_{\mathbb{S}_1\cup\mathbb{A}^n})}{\mathcal{V}(\mathcal{C}_{\mathbb{S}_1\cup\mathbb{OPT}^n})}.$$

Please note that for any fixed $n$, when $\mathbb{S}_1 = \emptyset$,the MAXINFO is exactly the weighted *Max $n$-Cover* problem [8].

*Theorem 1:* [8] If Max $n$-Cover can be constructively approximated in polynomial time within a ratio of $(1-1/e+\epsilon)$ for some $\epsilon > 0$, then $NP \subset TIME(p^{O(\log\log p)})$, where $p$ is the cardinality of the set (as $|\mathbb{S}_2|$ in Definition 4).

Theorem 1 directly implies that achieving a better approximation ratio than $(1 - 1/e)$ for MAXINFO is $NP$-hard. Thus, we have the following Corollary.

*Corollary 1 (Approximation Bound for* MAXINFO*):* Achieving approximation ratio $(1 - 1/e + \epsilon), \forall\epsilon > 0$ for MAXINFO is $NP$-hard.

### C. Greedy Algorithm

In this section, we outline our prioritization algorithm. The idea of the algorithm is to give higher transmission priority to data with larger marginal coverage value *w.r.t.* the dataset at the receiver side.

**Algorithm 1** Prioritization Algorithm

**Input:** Two sets $\mathbb{S}_1$ and $\mathbb{S}_2$
**Output:** An order of elements in $\mathbb{S}_2$

```
Set 𝕋 ← 𝕊1
FIFO Queue 𝒬 ← {}
while 𝕊2 ≠ ∅ do
    X ← arg max_{X∈𝕊2} 𝒱(ℳ𝒞_{X|𝕋})
    𝕋 ← 𝕋 ∪ {X}, 𝕊2 ← 𝕊2 − {X} , 𝒬.inqueue(X)
Return 𝒬
```

We now prove that the approximation ratio of Algorithm 1 is $(1 - \frac{1}{e})$.

*Lemma 1:* For any $n \leq |\mathbb{S}_2|$, if $\mathbb{R}$ is the set of the first $n$ elements of the queue output by Algorithm 1, we have

$$\mathcal{V}(\mathcal{C}_{\mathbb{S}_1\cup\mathbb{R}}) \geq (1 - 1/e) \cdot \mathcal{V}(\mathcal{C}_{\mathbb{S}_1\cup\mathbb{R}'}), \forall\mathbb{R}', |\mathbb{R}'| = |\mathbb{R}| = n,$$

where $\mathbb{S}_1$ (*resp.* $\mathbb{S}_2$) is the dataset at the data receiver (*resp.* provider) side.

The proof of Lemma 1 is similar as that in [8], except that in [8] $\mathbb{S}_1 = \emptyset$. We omit the proof here. Lemma 1 directly

implies the following theorem.

*Theorem 2 (Approximation Ratio):* The coverage value of the transmitted set based on the order output by Algorithm 1 is $(1-1/e)$-approximated for MAXINFO.

Note that the apprximation ratio matches the approximation bound in Corollary 1.

### D. Transmission Protocol Design

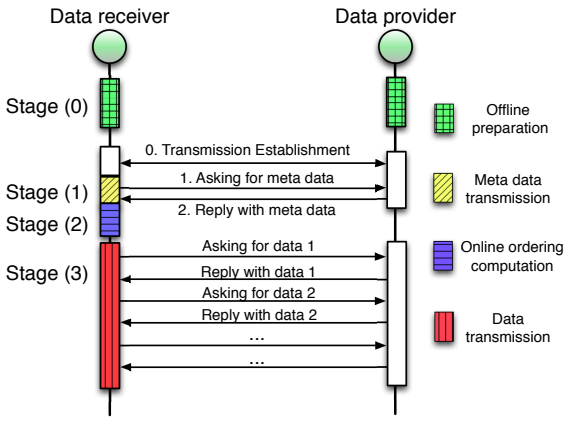



Fig. 4. Transmission protocol illustration.

In this section, we present the transmission protocol, as illustrated in Fig. 4. Since we target on mobile platform, the transmission is in a DTN fashion; a device shares its data with a peer or offloads to a backend server when the connection is established. Thus, each transmission session (in which case we call the device is *online*) is followed by an idle session (that we call the device *offline*).

Each transmission session contains three stages; (1) meta data transmission, (2) online ordering, and (3) data transmission. Stages (1) and (2) are the transmission overhead. Please note that stage (2) only do computation on the data receiver side, so the data receiver can also use this period of time to pull data from the data provider based on naive order, *e.g.*, FIFO. Before transmission, an offline preparation operation that generates the meta data needs to be carried out to reduce the overhead of the online ordering computation. We now present the offline preparation algorithm and online prioritization algorithm in detail as follows.

*1) Offline Preparation:* The offline preparation stage outputs a meta data file which contains a list of data names as well as the overlap set of each *heavily informative* data point as described below. (The overlap set of data $X$ contains any data $Y$ *s.t.* $\mathcal{C}_Y \cap \mathcal{C}_X \neq \emptyset$.)

Consider two data points $X$ and $Y$, if the coverage of $X$ greatly overlaps with that of $Y$, then after $X$ has been transmitted, $Y$ carries little extra information. Thus, we introduce a constant threshold $\beta > 1$, *s.t.*, when the distance between $X$ and $Y$ is smaller than $\frac{1}{\beta}$ in each dimension of the information space, we only need to consider any of them (*w.l.o.g.*, say $X$) in the online prioritization algorithm and directly assign the other data ($Y$) with the least priority. In this case, $X$ is thus called the heavily informative data point. The heavily informative dataset contains all the data points like $X$.

Our offline preparation algorithm is incremental in the sense that when new data points arrive we do not need to redo the preparation for old data. The offline preparation algorithm is presented in Algorithm 2.

**Algorithm 2** Preparation Algorithm

**Input:** Existing dataset $\mathbb{S}$, existing meta data file, newly arrived dataset $\mathbb{T}$

**Output:** Updated meta data file

1: From meta data, get the heavily informative dataset $\mathbb{H}$ of $\mathbb{S}$
2: Sort $\mathbb{H}$ based on the lexicographical order of data coordinates in the $k$-D information space
3: $\mathbb{D} \leftarrow \emptyset$, $\mathbb{N} \leftarrow \emptyset$
4: **for** $\forall S \in \mathbb{T}$ **do**
5: Use binary search to find its overlap set $\mathcal{O}_S \subseteq \mathbb{H}$
6: **if** $\exists E \in \mathcal{O}_S$, *s.t.* $S \simeq E$ **then**
7: $\mathbb{D} \leftarrow \mathbb{D} \cup \{S\}, \mathbb{T} \leftarrow \mathbb{T} - \{S\}$, **continue**
8: Add $S$ to the overlap set of any element in $\mathcal{O}_S$
9: Insert $S$ into $\mathbb{H}$ *s.t.* $\mathbb{H}$ remains sorted
10: $\mathbb{N} \leftarrow \mathbb{N} \cup \{S\}$
11: Add the name following by the overlap set of each data in $\mathbb{N}$ to the front of the meta data file
12: Append names of data in $\mathbb{D}$ to the end of meta data file
13: Return the meta data file

In our online ordering stage, we only need to consider the heavily informative dataset, which can be extracted from meta data, and transmit them first. The parameter $\beta$ controls the cardinality of the overlap set of a highly informative data. The smaller $\beta$ is, the smaller the cardinality of the overlap set is. In practical, we can use $\beta$ as a knob to trade-off the accuracy and the time efficiency in the online prioritization as discussed below.

*2) Online Prioritization:* After got the metadata, the receiver is ready to do the online prioritization which is described in Algorithm 3. In this algorithm, the receiver calculates the marginal coverage value of each data in the highly informative set got from the data provider, and put the these values into a max heap. Then, it sends the request of the data $D$ poped from the max heap, and at the same time update the marginal coverage value of each data in the overlap set of $D$ and do the standard heapify. This process continues until the max heap is empty, then, if the connection is still up, the receiver starts to pull data that not in the highly informative set in FIFO order.

*3) Overhead Analysis:* The time complexity of Algorithm 3 is $O(m \log n + mn^{\frac{k-1}{k}} + m\beta^k \log m)$, where $n$ is the number of highly informative data in $\mathbb{S}_1$, $m$ is the number of highly informative data in $\mathbb{S}_2$, and $\beta^k$ is the size bound of a overlap set as stated in the offline preparation section. In the worst case, $n = |\mathbb{S}_1|$ and $m = |\mathbb{S}_2|$, however both of them are related to the parameter $\beta$. Thus $\beta$ serves as a knob to trade-off prioritization accuracy and computational efficiency in our algorithm. We

**Algorithm 3** Online Prioritization Algorithm

**Input:** The highly informative dataset $\mathbb{S}_1$ on the receiver side which is sorted based on the lexicographical order of data coordinates in the $k$-D information space, meta data from the data provider

**Output:** The transmission order (represented by a FIFO queue $\mathcal{Q}$)

Initiate a Max-heap $\mathcal{H}$ that stores and sorts data points according to their associated values
**for** Each heavily informative $S$ in $\mathbb{S}_2$ **do**
  Use binary search to get its overlap set $\mathbb{T} \subseteq \mathbb{S}_1$
  Calculate $v_S = \mathcal{V}(\mathcal{MC}_{S|\mathbb{T}})$
  $\mathcal{H}$.add($S$, $v_S$)
Initiate a FIFO queue $\mathcal{Q}$
**while** $\mathcal{H} \neq \emptyset$ **do**
  $X \leftarrow \mathcal{H}$.popMax()
  {At the same time send request $X$ to the data provider.}
  $\mathcal{Q}$.enque($X$)
  For each data $S$ in $X$'s overlap set $\mathcal{O}_X$, update $v_S = \mathcal{V}(\mathcal{MC}_{S|\mathbb{T}\cup\mathcal{Q}})$, and update the heap $\mathcal{H}$
Append all other data in $\mathbb{S}_2$ to $\mathcal{Q}$
Return $\mathcal{Q}$

will further study the parameter $\beta$ in the overhead evaluation.

## V. Evaluation

In this section, we study the performance of Minerva. We first describe the experimental setup and implementation of Minerva on Android phones. Then we explain the experimental methodologies for both the real-phone based experiments and the simulations using the T-Drive dataset collected by MSRA. Evaluation results are presented at last.

### A. Experimental Setup and Implementation

To study the performance of Minerva in real world mobile social sensing applications, we implement Minerva on Google Galaxy Nexus smartphones [2], equipped with a 1.2 GHz dual-core CPU, 1GB RAM, and 802a/b/n WiFi radio. Minerva is implemented using the Java programming language on Android OS 4.1 on top of PARC's NDN prototype software [1] Please note that although NDN provides a perfect environment to implement Minerva, *it does not mean that Minerva can only run on top of NDN*. The reason is that the only feature of NDN used by Minerva is that "data have names". So Minerva can also be easily ported to TCP/IP networking by assigning each data point a key that serves as its name.

In Minerva, the `Map` function, as illustrated in Fig. 2, is implemented as a callback that is to be defined by applications to map data names to data position and coverage in the information space. The `Map` function is necessary since the semantics of data information is highly application specific. One simple example of name design is to embed the data features and coverage into its name, like `prefix/f1/int1/f2/int2/.../data-name`, where `f`$i$ is the coordinate in the $i$-th dimension and `int`$i$ is the coverage interval along the $i$-th dimension. The `Map` function in this case would be trivial. Application developers can define more sophisticated `Map` functions to meet their own specific application needs.

The prioritization engine (PE) of Minerva is implemented to be modular, allowing flexibility in choosing prioritization algorithms, such as distance-based prioritization algorithms or the trivial FIFO ordering. We will discuss the prioritization algorithms we use to compare with Minerva in the following methodology section. PE takes as input the returned values of `Map`, and computes an order based on which data are to be transmitted.

Minerva is designed as a special publish-subscribe framework, providing only two API's to application developers to simplify programming efforts. From our own experience, implementing a social sensing application that collects and shares traffic data following a information maximizing scheme requires ~1000 lines-of-code without the use of Minerva; the number of lines decreases to ~100 when we do take advantage of the API's Minerva provides, i.e., a 90% save on the programming effort. When `Pub` is called, it only puts the data to be published into a local repository and returns. The `Sub` function has only one parameter, the *interest* (name prefix) of the application data. When called, `Sub` only inserts the interest into a subscription queue and returns. Note that the prioritization of subscription interest is not the focus of this paper; we focus on prioritization the data transmission per subscription. When two phones have set up the connection, the queued interests will be sent to the other side, and data transmission takes place as illustrated in Fig. 4.

### B. Methodology

Minerva is designed for social sensing applications with transmission resource constraints, thus we need to evaluate two aspects: (1) the overhead of transmission in real-world scenarios, (2) the performance, measured as information coverage gain, of our algorithm in large-scale and real-world applications. The overhead study of Minerva is based on two Android smartphones running experiments in an out-door environment to simulate data sharing in real-world social sensing applications. To test the performance of Minerva in large-scale and real-world applications, we use the T-Drive dataset [3], [25], [26], which contains the GPS trajectories of $10,357$ taxis during the period from February 2nd to February 8th, 2008 in Beijing. The total number of points in this dataset is about $15$ million and the total distance of the trajectories is around 9 million kilometers.

We use synthetic data in our out-door phone-based experiments. The use of synthetic data is because we want to fully control the data to test the performance of Minerva under different scenarios as thoroughly as possible. We also study the performance of Minerva using the synthetic data under different settings, by varying (1) the number of data points, (2) the overlap probability, (3) the number of features, and (4) the trade-off parameter between accuracy and computational efficiency ($\beta$, introduced in Section IV D).

The synthetic data used in our experiment is generated within a unit $k$-D box (that we call the "universe"), where

$k$ is the number of features. All data points are generated uniformly at random in the universe. The reason why we use uniform generation is that we can easily control the overlap probability by the value of coverage interval length of each dimension. The coverage of data in our experiments is set to be the same.

We test the performance of Minerva in large-scale real-world applications through simulations on real taxi traces in Beijing (the T-Drive dataset). We consider data within the area from latitude $39.5^oN$ to $40.5^oN$ and from longitude $116^oE$ to $117^oE$, where most data locates. In the simulation, we assume there are two sinks that collect data for the backend server as shown as the two stars in Fig. 5. They are located in two relatively "busy" roads, where cars can have higher chance to offload their data. Cars are assumed to not share data with each other. We assume the wireless interference can be solved by existing methods, thus do not consider this issue in our paper. We also assume the data set collected by the two sinks are automatically and immediately synced.

In our simulation, if the distance between a car and a sink is smaller than 100 meters, the car can offload data. We assume that the driving data (location, speed, time, etc.) is collected by each car with the sample rate of 1Hz. 10 samples take roughly 1KB. In the T-Drive dataset, a car records one GPS location per 5 minutes, thus each data point contains 300 driving data samples. Therefore, each data sample in our simulation is roughly 30KB.

We determine the transmission duration of each car by examining its speed when it enters the transmission range of a sink; the speed can be estimated from the time and location information of the latest GPS samples.

In order to obtain a realistic WiFi transmission rate, we conduct a small experiment where we use a smartphone to send a 1.6MB picture over Wifi in an outdoor environment to a desktop in our lab. The average time to finish the transmission is 6.806 seconds, thus we set the data transmission rate to be 250KB per second in the simulation.

In our simulation, we consider two features per data sample, the latitude and the longitude. For different applications, the coverage interval for each dimension might be different. For example, if the application is to study the speed map of a city, each data point might cover 50 meters or 100 meters along each dimension. If the application is to take pictures of landmarks of the city, each data point might cover around 500 meters along each dimension. In our simulation, we consider the performance of Minerva under different coverage interval settings. We simulate for 10 hours' data (1,500,000 points) and assume at the very beginning the server does not have any data.

We compare Minerva with three other prioritization algorithms: (1) FIFO, (2) Random order, and (3) Distance based prioritization algorithm (*e.g.*the algorithm used in PhotoNet [20]). The distance based algorithm used in our study is *to transmit the data with the largest minimum distance to every data point on the receiver side in information space*. In other words, given a remote dataset $\mathbb{R}$ and a local dataset $\mathbb{L}$, we will always transmit $R \in \mathbb{R}$, such that $R = \arg\max \min dist(R, L), \forall L \in \mathbb{L}$. Please note that in our synthetic data for real-phone based experiments, we randomly generate the data, thus FIFO and Random ordering are the same.

[4]This figure is borrowed from [3].


Fig. 5. The data considered in the simulation.[4]

## C. Results

We first study the overhead of Minerva as described in Section IV using the synthetic data. The results are shown in Table. I.

The parameter set considered includes (1) the number of features, (2) the number of data points, (3) the coverage interval, and (4) the value of $1/\beta$. Rows 1-15 show the communication and computation overheads between two android phones. In this overhead study, data are generated using the same set of parameters on both the receiver side and the provider side. Rows 16-19 show only the computation overhead on a backend server with a 3.10GHz CPU and 8GB RAM.

TABLE I
OVERHEAD OF MINERVA

| index | dataset features (# dim, # points, interval, $1/\beta$) | overhead($s$) |
|---|---|---|
| 1 | 2 , **250** , 0.05 , 0.1 | $0.321 \pm 0.165$ |
| 2 | 2 , **500** , 0.05 , 0.1 | $0.837 \pm 0.208$ |
| 3 | 2 , **750** , 0.05 , 0.1 | $3.070 \pm 1.240$ |
| 4 | 2 , **1000** , 0.05 , 0.1 | $7.205 \pm 2.579$ |
| 5 | 2 , 500 , **0.03** , 0.1 | $0.582 \pm 0.104$ |
| 6 | 2 , 500 , **0.05** , 0.1 | $0.837 \pm 0.208$ |
| 7 | 2 , 500 , **0.07** , 0.1 | $1.667 \pm 0.320$ |
| 8 | 2 , 1000 , **0.03** , 0.1 | $2.213 \pm 0.848$ |
| 9 | 2 , 1000 , **0.05** , 0.1 | $7.205 \pm 2.579$ |
| 10 | 2 , 1000 , **0.07** , 0.1 | $17.082 \pm 4.510$ |
| 11 | 2 , 1000 , 0.05 , **0.15** | $5.591 \pm 2.310$ |
| 12 | 2 , 1000 , 0.05 , **0.2** | $3.060 \pm 1.203$ |
| 13 | **2** , 1000 , 0.1 , 0.2 | $3.580 \pm 1.335$ |
| 14 | **3** , 1000, 0.1 , 0.2 | $5.992 \pm 3.500$ |
| 15 | **4** , 1000, 0.1 , 0.2 | $1.706 \pm 0.815$ |
| 16 | 2, (10000, 1000), 0.01, 0.1 | $0.116 \pm 0.047$ |
| 17 | 2, (100000, 1000), 0.01, 0.1 | $2.387 \pm 0.232$ |
| 18 | 2, (1000000, 1000), 0.01, 0.1 | $8.431 \pm 1.357$ |
| 19 | 2, (1000000, 1000), 0.01, 0.2 | $1.468 \pm 0.117$ |
| 20 | Wifi connection establish time | $2.002 \pm 0.106$ |

From the table, we observe that the overhead grows with the number of data points and the coverage interval, but decreases

as $1/\beta$ increases, which corroborates our overhead analysis in Section IV.

It is hard to tell how the number of features affects the overhead as shown from Row 13 to 15 in the table. With 1000 data points, the coverage interval set to 0.1 and $1/\beta$ to 0.2, the overhead in a 2-D space is smaller than that in a 3-D space, but larger than that in a 4-D space. The reason is in the 4-D space with 1000 data points, the chance that the coverage regions of two points overlap with each other is small. Thus the online ordering computation is greatly simplified. When the overlap possibility is equal under different numbers of features, the case with more features would have a higher computation overhead, as reflected from Row 8 and Row 14; in both cases, the expected cardinality of the overlap set of one data point is 1.

The table also shows that under certain settings the overhead is above 10 seconds which is unacceptable in opportunistic data sharing when the bottleneck resource is the short length of communication session. However, please note that most of the overhead is due to online computation, since the meta data is only a few kilo bytes or tens of kilo bytes which only takes less than 1 second to transmit. During the online computation, the receiver can pull data from the provider in a FIFO order. When the computation is finished, the receiver pulls the rest of the data based on the computed order.[5] In conclusion, the online computation does not block data transmission; the receiver can always pull data from the data provider in parallel with the computation. In other words, the overhead of Minerva is acceptable under all practical scenarios.

The computation overheads when data is being offloaded from mobile device to the backend server are shown in Row 16 to 18. The number of data points is set to be 1000 on the participant side, and the number of data points on server side is set to be $10\times$, $100\times$, and $1000\times$ of that on participant side. We observe that with the number of data points increasing on the server side, the computation time grows. We can also observe that the slope of overhead increase becomes smaller when the number of data points at the server side becomes larger.The reason is that the coverage gain grows submodularly; when the server already got a large enough amount of data, the probability that a new data point is redundant is close to 1, therefore we do not need to consider it in the ordering computation.

Next, we study the performance of Minerva between two smartphones using the synthetic data.The number of data points is set to 500 on both phones, and the coverage interval is set to be 0.063, 0.077, and 0.089 (thus, in expectation, one data point overlaps with 2, 3, and 4 other points respectively). For each interval value, we plot the coverage gain using Minerva (with $1/\beta \in \{0.05, 0.1, 0.15, 0.2\}$), Dist and FIFO. The x-axis is the time in milliseconds that starts right after the connection is established. The y-axis denotes the normalized information coverage gain. The results are plotted in Fig. 6.

[5]A bloom filter can be used to check whether or not the data is pulled already with constant time.

From Fig. 6 we observe that Minerva outperforms the other two algorithms in general. The larger the coverage interval is,the better Minerva performs, since the probability that two data points are redundant is higher. From the figure, to gain 80% information coverage, Minerva uses 10 (8 and 5 *resp.*) seconds with the coverage interval of 0.063 (0.077 and 0.089 *resp.*). Dist uses around 20 seconds to gain 80% information coverage, while FIFO takes more than 50 seconds.

From Fig. 6, we can observe that when $1\beta = 0.2$, Minerva produces the smallest overhead, but it does not always transmit data with the largest marginal coverage, which also corroborates our analysis in Section IV.

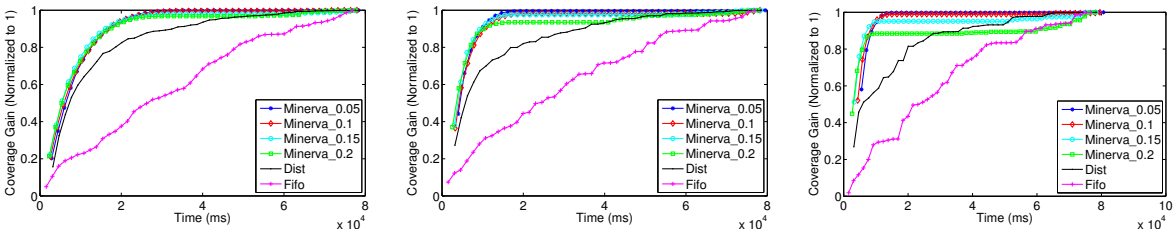
Finally, we study the performance of Minerva in large-scale real-world applications through simulations using the T-Drive dataset. The results are shown in Fig. 7. From Fig. 7, we observe that at the beginning of the data collection, the four algorithms yield similar performance. Minerva is slightly worse than the others due to its overhead. After collecting data for one hour, Minerva begins to outperform the others. The reason is that Minerva is designed for eliminating redundant data, but at the beginning there is little redundancy since the server has only a limited amount of data. For the same reason, from Fig. 7, we can also observe that as the coverage interval increases, data redundancy becomes higher, the performance of Minerva also improves.

Fig. 7 also shows that the information coverage gain grows faster during 7am to 9am and during 4pm to 6pm than other times, due to the rush hour natures of these two time intervals. So the speed of cars is smaller during these times, thus transmission duration is longer.

After 10 hours simulation, the information coverage gain by Minerva increases 17% (23% and 25% *resp.*) with coverage interval of 50 meters (100 meters and 500 meters *resp.*) compared with the distance-based algorithm, increases 23% (37% and 44% *resp.*) with that of 50 meters (100 meters and 500 meters *resp.*) compared with the Random ordering, and increases 33% (46%, and 53% *resp.*) with the same coverage intervals compared with FIFO.

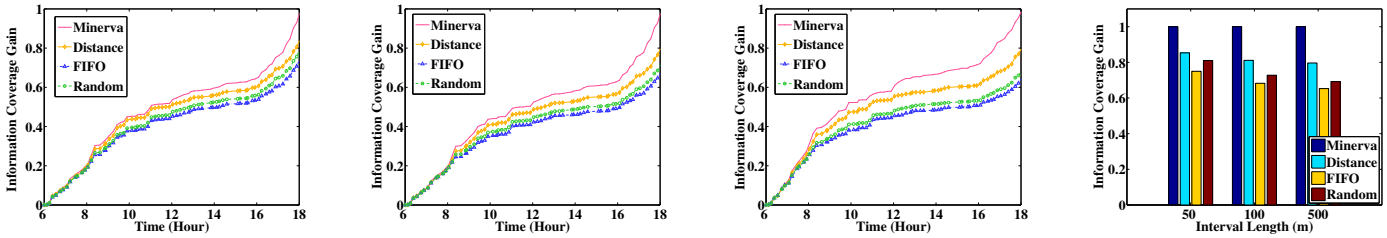
## VI. Conclusion

In this paper, we present Minerva; an information-centric programming paradigm and toolkit for social sensing. To the best of our knowledge, this is the first paper investigating the benefit of NDN in social sensing applications to maximize information coverage. Minerva is geared for social sensing applications, where different sources (participants sharing sensor data) often overlap in information they share, which distinguishes from a regular publish-subscribe system where publishers are assumed to be independent. One contribution in this paper lies in an algorithm for maximizing information delivery from publishers to subscribers taking into account the *non-independent* nature of content. We analytically prove that our transmission prioritization scheme is constant approximated compared with offline optimal solution. We develop a toolkit to simplify programming significantly for social sensing applications by separating application-specific functions




(a) Coverage interval 0.063 (b) Coverage interval 0.077 (c) Coverage interval 0.089

Fig. 6. Performance of Minerva with different coverage intervals and $1/\beta$ values in phone-based experiments with synthetic data.




(a) Coverage interval 50m (b) Coverage interval 100m (c) Coverage interval 500m (d) Final coverage gain

Fig. 7. Performance of Minerva with different coverage intervals in large-scale real-world GPS trace simulations.

from application-independent ones. Evaluation results show that our algorithm outperforms other candidate algorithms by up to 53% in terms of information coverage.

## REFERENCES


[1] CCNx prototype software. `http://www.ccnx.org`.

[2] Google galaxy nexus. `http://www.google.com/nexus`.

[3] User guide of t-drive data. `http://research.microsoft.com/pubs/152883/User_guide_T-drive.pdf`.

[4] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich. Mobiscopes for human spaces. *Pervasive Computing, IEEE*, 6(2):20–29, 2007.

[5] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker. Packet caches on routers: the implications of universal redundant traffic elimination. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 219–230. ACM, 2008.

[6] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and MB Srivastava. Participatory sensing. 2006.

[7] S.B. Eisenman, E. Miluzzo, N.D. Lane, R.A. Peterson, G.S. Ahn, and A.T. Campbell. Bikenet: A mobile sensing system for cyclist experience mapping. *ACM Transactions on Sensor Networks (TOSN)*, 6(1):6, 2009.

[8] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[9] R.K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T.F. Abdelzaher. Greengps: A participatory sensing fuel-efficient maps application. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 151–164. ACM, 2010.

[10] R.K. Ganti, N. Pham, Y.E. Tsai, and T.F. Abdelzaher. Poolview: stream privacy for grassroots participatory sensing. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 281–294. ACM, 2008.

[11] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker. Naming in content-oriented architectures. In *Proc of SIGCOMM Workshop on ICN*, 2011.

[12] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Information-centric networking: seeing the forest for the trees. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, page 1. ACM, 2011.

[13] J.H. Huang, S. Amjad, and S. Mishra. Cenwits: a sensor-based loosely coupled search and rescue system using witnesses. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 180–191. ACM, 2005.

[14] J.H. Huang, S. Amjad, and S. Mishra. Cenwits: a sensor-based loosely coupled search and rescue system using witnesses. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 180–191. ACM, 2005.

[15] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, and R.L. Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM, 2009.

[16] H. Liu, A. Srinivasan, K. Whitehouse, and J.A. Stankovic. Mélange: Supporting heterogeneous qos requirements in delay tolerant sensor networks. In *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*, pages 93–96. IEEE, 2010.

[17] M. Meisel, V. Pappas, and L. Zhang. Ad hoc networking via named data. In *Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture*, pages 3–8. ACM, 2010.

[18] M. Squirrel. Web cache control.

[19] G. Tyson, N. Sastry, I. Rimac, R. Cuevas, and A. Mauthe. A survey of mobility in information-centric networks: challenges and research directions. In *Proceedings of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design-Architecture, Algorithms, and Applications*, pages 1–6. ACM, 2012.

[20] M.Y.S. Uddin, H. Wang, F. Saremi, G.J. Qi, T. Abdelzaher, and T. Huang. Photonet: A similarity-aware picture delivery service for situation awareness. In *Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd*, pages 317–326. IEEE, 2011.

[21] B. Wang. Coverage problems in sensor networks: A survey. *ACM Computing Surveys (CSUR)*, 43(4):32, 2011.

[22] L. Wang, R. Wakikawa, R. Kuntz, R. Vuyyuru, and L. Zhang. Data naming in vehicle-to-vehicle communications.

[23] U. Weinsberg, A. Balachandran, N. Taft, G. Iannaccone, V. Sekar, and S. Seshan. Care: Content aware redundancy elimination for disaster communications on damaged networks. *Arxiv preprint arXiv:1206.1815*, 2012.

[24] A. Wolman, M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H.M. Levy. On the scale and performance of cooperative web proxy caching. *ACM SIGOPS Operating Systems Review*, 33(5):16–31, 1999.

[25] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 316–324. ACM, 2011.

[26] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 99–108. ACM, 2010.

[27] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J.D. Thornton, D.K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, et al. Named data networking (ndn) project. Technical report, PARC, Tech. report ndn-0001, 2010.